Parallel Computing Basics, Semantics

Landau's 1st Rule of Education

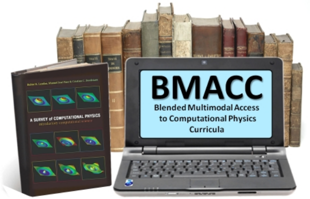
Rubin H Landau

Sally Haerer, Producer-Director

Based on *A Survey of Computational Physics* by Landau, Páez, & Bordeianu

with Support from the National Science Foundation

Course: **Computational Physics II**



Parallel Problems

Basic and Assigned

- Impressive parallel (‖) computing hardware advances
- Beyond ‖ I/O, memory, internal CPU
- ‖: multiple processors, single problem
- Software stuck in 1960s
- Message passing = dominant, = too elementary
- Need sophisticated compilers (OK cores)
- Understanding hybrid programming models
- **Problem:** Parallelize simple program's parameter space
- Why do? faster, bigger, finer resolutions, different

$\|$ Computation Example, Matrix Multiplication

Need Communication, Synchronization, Math

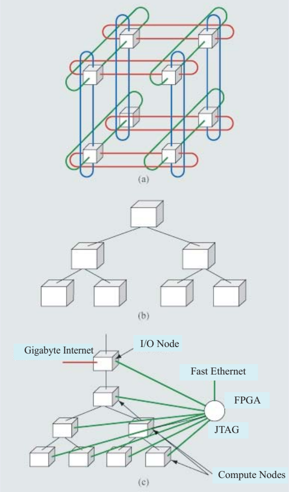
$$[B] = [A][B] \tag{1}$$

$$B_{i,j} = \sum_{k=1}^{N} A_{i,k} B_{k,j} \tag{2}$$

- Each LHS $B_{i,j}$ $\|$
- Each LHS row, column $[B]$ $\|$
- RHS $B_{k,j}$ = old, before mult values $\Rightarrow$ communicate
- $[B] = [A][B]$ = data dependency, order matters
- $[C] = [A][B]$ = data parallel

Parallel Computer Categories

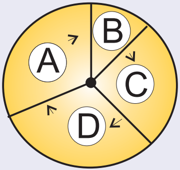
Nodes, Communications, Instructions & Data

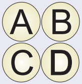


- CPU-CPU, mem-mem networks
- Internal (2) & external
- Node = processor location
- Node: 1-N CPUs
- Single-instruction, single-data
- Single-instruction, multiple-data
- Multiple instructs, multiple data
- MIMD: message-passing
- MIMD: no shared mem cluster
- MIMD: Difficult program, $

Relation to MultiTasking

Locations in Memory (s)

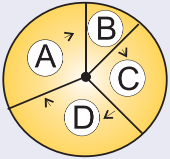


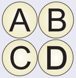


- Much $\|$ on PC, Unix
- Multitasking $\sim \|$
- Indep progs simultaneously in RAM
- Round robin processing
- SISD: 1 job/t
- MIMD: multi jobs/same t

Parallel Categories

Granularity





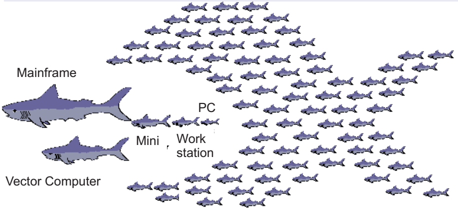
- Grain = measure computational work
- = computation / communication

- **Coarse-grain:** Separate programs & computers
- e.g. MC on 6 Linux PCs
- **Medium-grain:** Several simultaneous processors
- Bus = communication channel
- Parallel subroutines Δ CPUs
- **Fine-grain:** custom compiler
- e.g. || `for` loops

Distributed Memory || via Commodity PCs

Clusters, Multicomputers, Beowulf, David

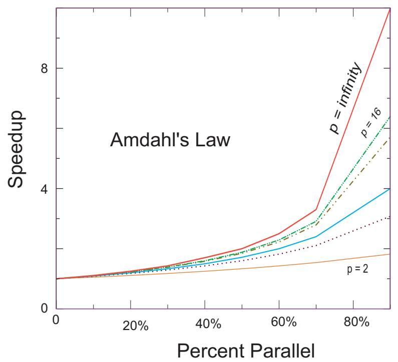
Values of Parallel Processing



- Dominant coarse-medium grain
- = Stand-alone PCs, hi-speed switch, messages & network
- Req: data chunks to indep busy ea processor
- Send data to nodes, collect, exchange, ...

Parallel Performance: Amdahl's law

Simple Accounting of Time



- Clogged ketchup bottle in cafeteria line
- Slowest step determines reaction rate
- $\|$ serial, communication = ketchup
- Need ∼90% parallel
- Need ∼100% for massive
- Need new problems

Amdahl's Law Derivation

$$p = \text{no. of CPUs} \quad T_1 = \text{1-CPU time}, \quad T_p = p\text{-CPU time} \tag{1}$$

$$S_p = \text{max parallel speedup} = \frac{T_1}{T_p} \to p \tag{2}$$

- Not achieved: some serial, data & memory conflicts
- Communication, synchronization of the processors
- $f = \|$ fraction of program $\Rightarrow$

$$T_s = (1-f)\,T_1 \quad \text{(serial time)} \tag{3}$$

$$T_p = f\,\frac{T_1}{p} \quad \text{(parallel time)} \tag{4}$$

$$\text{Speedup } S_p = \frac{T_1}{T_s + T_p} = \frac{1}{1 - f + f/p} \quad \text{(Amdahl's law)} \tag{5}$$

Amdah's Law + Communication Overhead

Include Communication Time; Simple & Profound

- Latency = T_c = time to move data

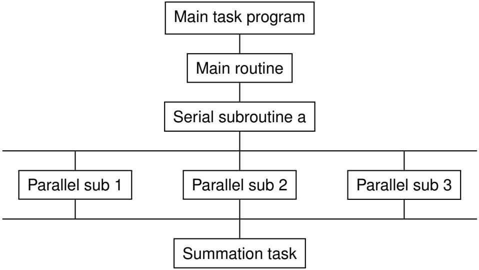
$$S_p \simeq \frac{T_1}{T_1/p + T_c} < p \tag{1}$$

- For communication time not to matter

$$\frac{T_1}{p} \gg T_c \quad \Rightarrow \quad p \ll \frac{T_1}{T_c} \tag{2}$$

- As $\uparrow$ number processors p, $T_1/p \rightarrow T_c$
- Then, more processors $\Rightarrow$ slower
- Faster CPU irrelevant

How Actually Parallelize



- User creates tasks
- Task assigns processor threads
- Main: master, controller
- Subtasks: parallel subroutines, slaves

- Avoid storage conflicts
- $\downarrow$ Communication, synchronization
- Don't sacrifice science to speed

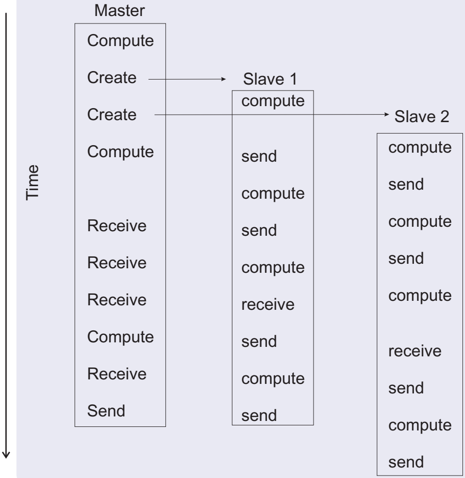
Practical Aspects of Message Passing; Don't Do It

More Processors = More Challenge

- Only most numerically intensive $\|$
- Legacy codes often Fortran90
- Rewrite (N months) *vs* Modify serial ($\sim$70%)?
- Steep learning curve, failures, hard debugging
- Preconditions: run often, for days, little change
- Need higher resolution, more bodies
- **Problem affects parallelism:** data use, problem structure
- Perfectly (embarrassingly) parallel: (MC) repeats
- Fully synchronous: Data $\|$ (MD), tightly coupled
- Loosely synchronous: (groundwater diffusion)
- Pipeline parallel: (data $\rightarrow$ images $\rightarrow$ animations)

High-Level View of Message Passing

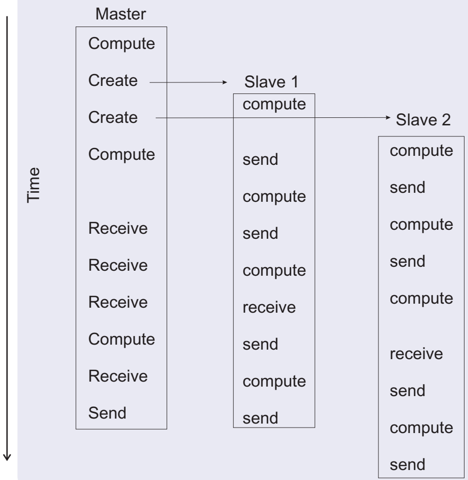
4 Simple Communication Commands



- Simple basics
- C, Fortran + 4 communications
- **send:** named message
- **receive:** any sender
- **myid:** ID processor
- **numnodes**

|| MP: What Can Go Wrong?

Hardware Communication = Problematic



- Task cooperation, division
- Correct data division
- Many low-level details
- Distributed error messages
- Wrong messages order
- Race conditions: order dependent
- Deadlock: wait forever

Conclude: IBM Blue Gene = $\|$ by Committee

- Performance/watt
- On, off chip mem
- 2 core CPU
- 1 Core compute, 1 communicate
- 65,536 (2^{16}) nodes

- Peak = 360 teraflops (10^{12})
- Medium speed CPU
- 5.6 Gflop (cool)
- 512 chips/card, 16 cards/Board
- Control: MPI